Electronic Colloquium on Computational Complexity, Report No. 58 (2001)

A Note on the Minimum Number of Negations Leading to Superpolynomial Savings

Stasys Jukna *

Abstract

In 1957 Markov proved that every circuit in n variables can be simulated by a circuit with at most $\lceil \log(n+1) \rceil$ negations. In 1974 Fischer has shown that this can be done with only polynomial increase in size. In this note we observe that some explicit monotone functions in P cannot be computed by a circuit of polynomial size if we allow only $\log n - O(\log \log n)$ negations.

1 Introduction

We consider Boolean circuits with And, Or and Not gates. Such a circuit G is just a sequence $g_1, \ldots, g_t$ of Boolean functions $g_i : \{0,1\}^n \to \{0,1\}$ where each g_i is either one of the input variables $x_1, \ldots, x_n$ or is obtained from the previous functions by applying an Or, And or Not operation (such applications are called *gates*). The length t of this sequence is the *size* of the circuit. Given a multi-output function $f : \{0,1\}^n \to \{0,1\}^k$, we can consider it as the sequence $f_1, \ldots, f_k$ of Boolean (i.e., single-output) functions such that, for every input $a \in \{0,1\}^n$, $f_i(a)$ is the i-th bit of the output $f(a)$. A circuit computes such a function f if all the functions $f_1, \ldots, f_k$ belong to it.

*Universität Frankfurt, Institut für Informatik, Robert-Mayer-Str. 11-15, D-60054 Frankfurt, Germany & Institute of Mathematics and Informatics, Akademijos 4, LT-2600 Vilnius, Lithuania (jukna@thi.informatik.uni-frankfurt.de)

ISSN 1433-8092

The main difficulty in proving non-trivial lower bounds on the size of circuits is the presence of Not gates. More than 30 years ago, Markov [6] has made an intriguing observation that every function in n variables can be computed by a circuit with only $\lceil \log(n+1) \rceil$ negations.[1] Fischer [4] extended this result by showing that every circuit in n variables can be simulated by a circuit with only $\lceil \log(n+1) \rceil$ negations in such a way that the increase in size is only polynomial. Hence, if one could show that some function in NP cannot be computed by polynomial size circuit with that number of negations, this would imply $P \neq NP$. It is therefore natural to investigate the power of circuits with a limited number of negations.

In fact, Markov [6] gives a surprisingly tight connection between the number of necessary negations and so-called "decrease" of the function. For two binary vectors $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ write $x \leq y$ if $x_i \leq y_i$ for all i. Write also $x < y$ if $x \leq y$ and $x_i < y_i$ for at least one i. A *chain* in the binary n-cube is an increasing sequence $y^1 < y^2 < \cdots < y^s$ of vectors. The *decrease* of a function $f : \{0,1\}^n \to \{0,1\}^k$ *on a chain* Y is the number of indices i such that $f(y^i) > f(y^{i+1})$. The *decrease* $d(f)$ of f is the maximum its decrease over all chains. Note that $0 \leq d(f) \leq n$ for every function f in n variables, [2] and $d(f) = 0$ if and only if f is monotone, i.e., if $x \leq y$ implies $f(x) \leq f(y)$.

Theorem 1 (Markov 1957) *For every function f, the minimum number of Not gates contained in a circuit computing f is precisely $\lceil \log(d(f)+1) \rceil$.*

This result motivates the following question. Suppose that a function f in n variables *can* be computed by a circuit of size polynomial in n (in this case we say that f is *feasable*), but every circuit with $M(f) := \lceil \log(d(f)+1) \rceil$ negations computing f requires superpolynomial size. What is then the minimal number $R(f)$ of negations sufficient to compute f in polynomial size? Fischer's result only implies that this number lies somewhere between the Markov's lower bound $M(f)$ and the generic upper bound $\lceil \log(n+1) \rceil$. The discrepancy between these two bounds is especially evident for functions with $M(f) = 0$, i.e., for monotone functions. Berkowitz [3] and Valiant [13] have shown that for so-called *slice* functions (these are monotone Boolean functions which are non-trivial only on one slice of the n-cube) negations

[1] All logarithms are base two; hence, $\lceil \log(n+1) \rceil$ is the number of bits in the binary representation of n.

[2] And $0 \leq d(f) \leq \lfloor (n+1)/2 \rfloor$ for every *Boolean* function f in n variables.

are powerless, i.e., cannot lead to a superpolynomial savings. So, the first natural question was whether using Not gates can lead to superpolynomial savings in size *at all*?

This (long standing) problem was resolved by Razborov [9]: there exists an explicit feasable monotone Boolean function f (corresponding to the perfect matching problem in bipartite graphs) such that $R(f) > 0$. Another (less natural but still explcit) function, for which negations even yield exponential savings, was later given by Tardos [12]. The next natural question was: *how large does* $R(f)$ *actually is*? Or, in other words:

> How many Not gates do we actually need to achieve superpolynomial savings in circuit size?

This last question was considered by many authors under additional restrictions on the topology and/or on the use of Not gates. In particular, Okolnishnikova [7], and Ajtai and Gurevich [1] have shown that there exists monotone functions that can be computed with polynomial size, *constant* depth circuits, but can not be computed with *monotone*, polynomial size, constant depth circuits. This implies that $R(f) > 0$ in the class of AC^0 circuits. Moreover, it was shown by Satha and Wilson [11] that in this class we need much more than $\lceil \log(n+1) \rceil$ negations: there is a (multi-output) function computable in constant depth that cannot be computed in constant depth with $o(n/\log^{1+\epsilon} n)$ negations. (Note that this result does not contradict with the Markov–Fischer upper bound: their simulation requires logarithmic depth.) Another line of research was to restrict the use of Not gates. For circuits of logarithmic depth, a lower bound $R(f) = \Omega(n)$ was proved by Raz and Wigderson [8] under the restriction that *all* the negations are placed on the input variables: there is an explicit monotone function (corresponding to the connectivity problem for graphs) that can be computed with polynomial size, depth $O(\log^2 n)$ circuits, but can not be computed with polynomial size, depth $k \log n$ circuits using only $o(n/2^k)$ negated variables.

But it remained unclear whether there exist feasable monotone functions f for which $R(f) = \omega(1)$, if we put restrictions neither on the depth nor on the use of Not gates. A step in this direction was made by Amano and Maruoka [2] who proved that the clique function cannot be computed in polynomial size using only $(1/6) \log \log n$ negations. Unfortunately, this does not answer the question, because the clique function itself is not feasable (unless $P = NP$).

In this note we observe that, in fact, the results of Razborov and Tardos can be used to move the threshold $R(f)$ (of the first superpolynomial decrease in size) quite near to the Markov–Fischer bound $\lceil \log(n+1) \rceil$: there are (explicit) feasable monotone functions $f : \{0,1\}^n \to \{0,1\}^n$ such that $R(f) \geq \log n - c \log\log n$.

2 Feasable functions requiring many Not gates

Let $h = h(X)$ be a Boolean function in m variables $X = \{x_1, \ldots, x_m\}$. Let $k = 2^r$ and $n = km$. A function $f : \{0,1\}^n \to \{0,1\}^k$ is an *r-fold extension* of h if it computes k copies of h on disjoint copies $X_1, \ldots, X_k$ of X. That is, given an input $(a^1, \ldots, a^k)$ with $a^i \in \{0,1\}^{X_i}$, the function outputs the sequence $\Big(h(a^1), \ldots, h(a^k)\Big)$. Important here is that the i-th output bit $h(a^i)$ is independent of inputs a^j for $j \neq i$. Note also that the extension of monotone functions are monotone. A *minterm* of a monotone Boolean function is a minimal set of variables which, if assigned the value 1, forces the function to take the value 1 regardless of the values assigned to the remaining variables.

Lemma 1 *Let h be a monotone Boolean function, and r be a nonnegative integer. If h cannot be computed by a monotone circuit of size t, then its r-fold extension f cannot be computed by a circuit of size t using r negations.*

Proof. Given a circuit G which computes f and has r Not gates, we eliminate these gates one-by-one. To do this, consider the *first* Not gate in G, and let g be the monotone(!) Boolean function computed at the input to this gate (i.e., immediately before this gate). Let Y_0 and Y_1 be the union of the first 2^{r-1} and, respectively, the last 2^{r-1} blocks of variables $X_1, \ldots, X_{2^r}$. Let $f_0(Y_0)$ and $f_1(Y_1)$ be the corresponding $(r-1)$-fold extensions of h.

Since g is monotone, all its minterms are positive (no negated literals). Hence, we have only two possibilities: either some minterm of g lies entirely in Y_1, or not. In the first case we assign constant 1 to all the variables in Y_1, whereas in the second case we assign constant 0 to all the variables in Y_0. As the function g is monotone, in both cases it turns to a constant function (constant 1 in the first case, and constant 0 in the second), and the subsequent Not gate can be eliminated. Since $Y_0 \cap Y_1 = \emptyset$, the setting $Y_\epsilon \mapsto \epsilon$ does not affect the function $f_{1-\epsilon}$. Hence, we obtain a circuit which computes

an $(r-1)$-fold extension of h, and has one Not gate fewer. Repeating this argument r times we will obtain a circuit of the same (or smaller) size which computes h and has no Not gates. □

As a corollary we obtain

Theorem 2 *There exist explicit feasable monotone functions* $f_n : \{0,1\}^n \to \{0,1\}^n$ *such that* $R(f_n) \geq \log n - 9 \log\log n$.

Proof. Take an explicit monotone Boolean function T_m in m variables considered in [12]. As shown in [12], this function is feasable (can be computed by a non-monotone circuit of size $m^{O(1)}$) but every monotone circuit computing it requires size exponential in $\Omega(m^{1/8})$. Let $n = 2^r m$ and consider the r-fold extension f_n of T_m. The function f_n is feasable but, by Lemma 1, every circuit with at most r Not gates computing f_n must have size exponential in $\Omega(m^{1/8}) = \Omega\left((2^{-r}n)^{1/8}\right) = \Omega\left((\log n)^{9/8}\right)$, as long as $r \leq \log \frac{n}{(\log n)^9}$. Thus, $R(f_n) > r \geq \log n - 9\log\log n$. □

One may ask what happens if besides Not gates we will allow, say, monotone *real-valued* functions $\varphi : \mathsf{R}^2 \to \mathsf{R}$ as gates—does then the use of negations can still lead to a drastical decrease of size? The question makes sense because it is known (see [10]) that there exist a lot of monotone Boolean functions (slice functions) that can be computed by monotone real circuits (no Not gates) of linear size $O(n)$, but require Boolean circuits with And, Or and Not gates of size $2^{\Omega(n)}$. On the other hand, it is shown in [5] that Tardos' function T_m requires also monotone real circuits of size $2^{\Omega(m^{1/8})}$. This means that the function f_n considered in Theorem 2 captures the role of negations in a quite strong sense: this function is feasable but cannot be computed by a circuit of polynomial size with fewer than $\log n - 9\log\log n$ Not gates, even if we allow *all* monotone real-valued functions as gates.

The overall conclusion of this note is that, in the context of the P versus NP problem, it is important to understand the role of Not gates when their number r is *indeed* very close to the Markov–Fischer upper bound of $\lceil \log(n+1) \rceil$: moving by more than $\log\log n$ apart from this bound we esentially move by $\log n$ and reach the world of monotone circuits ($r = 0$) where Not gates play no role at all. It would be interesting to konw whether the same also holds for *single-output* functions.

References

[1] M. Ajtai and Y. Gurevich (1987): Monotone versus positive, *J. of the ACM* **34**, 1004–1015.

[2] K. Amano and A. Maruoka (1988): A superpolynomial lower bound for a circuit computing the clique function with at most $(1/6)\log\log n$ negation gates, in: Springer Lect. Notes in Comput. Sci., vol. 1450, 399–408.

[3] S.J. Berkowitz (1982): On some relationships between monotone and non-monotone circuit complexity. Technical Report, University of Toronto.

[4] M.J. Fischer (1974): The complexity of negation-limited networks–a brief survey, in: Springer Lect. Notes in Comput. Sci., vol. 33, 71–82.

[5] S. Jukna (1999): Combinatorics of monotone computations, *Combinatorica* **19**:1, 65–85.

[6] A.A. Markov (1957): On the inversion complexity of systems of Boolean functions, *Doklady Academii Nauk SSSR*, **116**, 917–919 (in Russian). English translation in: *J. of ACM*, **5**:4 (1958), 331–334, and in: *Soviet Math. Doklady* **4** (1963), 694–696.

[7] E.A. Okolnishnikova (1982): On the influence of negation on the complexity of realization of monotone Boolean functions by formulas of bounded depth, in: *Metody Diskretnogo Analiza* **38**, 74–80 (in Russian)

[8] R. Raz and A. Wigderson (1989): Probabilistic communication complexity of Boolean relations, in: Proc. of 30th Ann. IEEE Symp. on Foundations of Comput. Sci., 562–567.

[9] A.A. Razborov (1985): A lower bound on the monotone network complexity of the logical permanent, *Matematicheskie Zametki*, **37**:6, 887–990 (in Russian). English translation in: Math. Notes Acad. of Sci. USSR, **37**:6 (1985), 485–493.

[10] A. Rosenbloom (1997): Monotone real circuits are more powerful than monotone Boolean circuits, *Information Processing Letters* **61**, 161–164.

[11] M. Santha and Ch. Wilson (1993): Limiting negations in constant depth circuits, *SIAM J. Comput.* **22**:2, 294–302.

[12] É. Tardos (1987): The gap between monotone and non-monotone circuit complexity is exponential, *Combinatorica*, **7**:4, 141–142.

[13] L.G. Valiant (1986): Negation is powerless for Boolean slice functions, *SIAM J. Comput.* **15**, 531–535.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'